Categories, Allegories and Circuit Design

Carolyn Brown
COGS, University of Sussex
Brighton, BN1 9QH, UK
carolynb@cogs.susx.ac.uk

Graham Hutton
Chalmers University of Technology
Göteborg, Sweden
graham@cs.chalmers.se

Abstract

Relational languages such as RUBY *are used to derive hardware circuits from abstract specifications of their behaviour. Much reasoning is done informally in* RUBY *using pictorial representations of relational terms. We formalise this use of pictures in circuit design. We show that pictures naturally form a* unitary pretabular allegory*. Homomorphisms of pictures correspond to adding new wires or circuit components. Two pictures are mutually homomorphic if and only if they represent equal allegorical terms. We prove soundness and completeness results which guarantee that deriving circuits using pictures does not lead to errors. We illustrate the use of pictures by deriving the ripple adder implementation from a high level, behavioural specification.*

1: Introduction

Hardware circuit design involves translating abstract specifications of programs into efficient circuits which compute those programs. Pictures are widely used as an informal means of translating a specification into an implementable circuit, and improving the layout of a circuit. A disadvantage of this informal approach is the lack of an independent means (apart from building and testing the circuit, which may be expensive and is not guaranteed to succeed) to verify that a picture indeed denotes the desired program, and that no errors have been introduced during the design process.

In this paper we provide a relational semantics for pictures, together with an equivalence on pictures which shows how to transform one picture into another while preserving its semantics. The evident notion of homomorphism between pictures corresponds naturally to simple, relatively high level operations on pictures (adding new wires and components). Two pictures are provably equivalent if and only if they are mutually homomorphic, which is if and only if they denote the same relation for any interpretation of their basic components. These results lead us to a simple decision procedure for equivalence of circuits [3], which has been implemented [9].

Our results encourage the use of pictures in deriving circuits, by providing a formal foundation for that use. Pictures are easier and quicker to understand than syntactic terms, and so their use speeds up the process of circuit design. This paper illustrates these advantages in two ways, by illuminating the rather technical proof that the category of unitary pretabular allegories is isomorphic to the category of discrete cartesian bicategories, and by presenting the derivation of a ripple adder from a high level behavioural specification.

Relations have been proposed as a paradigm for circuit development for several reasons. Relations provide a rich algebra for transforming and combining terms, and a natural treatment of non-determinism. Furthermore, in practice many methods for combining functions in networks are unified if the distinction between input and output is relaxed [14], and many specifications can be expressed very naturally as representation-changers [10], that is, as the relational composition of a function with the converse of a function. Jones and Sheeran's relational calculus RUBY has been used to derive various kinds of hardware circuit from abstract behavioural specifications: examples include systolic arrays [15], butterfly networks [16] and arithmetic circuits [8, 10]. Other relational languages under development include that of Backhouse *et al* [1] and Bird and de Moor [2, 12]. Since these languages have the same underlying algebraic structure as RUBY, our techniques also apply to them.

Much informal reasoning in RUBY depends on a pictorial interpretation of a term as a network of primitive relations, and this pictorial interpretation is crucial when RUBY is used to develop circuit layouts. This paper formalises the pictorial interpretation of RUBY terms and algebraic laws, which has not previously been made precise.

Allegories abstract the notion of sets and relations rather as categories abstract the notion of sets and functions. We view pictures of circuits as arrows in a unitary pretabular allegory (upa) [5].This approach provides combinators for pictures corresponding to the operators of relational algebra, together with combinators corresponding to local products and projections. The equational axiomatisation of upas provides a notion of equivalence on pictures of circuits which cor-

responds to behavioural equivalence, in the sense that equivalent pictures represent equal relations.

Pictures are an excellent aid to reasoning but they are unwieldy and difficult to automate: we therefore introduce *networks* of wires and basic components. Networks abstract from the net list model of circuit connectivity [6, 13] by ignoring the size and position of components in a circuit. In the absence of empty types, the allegory of networks and network homomorphisms is equivalent to the allegory of pictures and picture homomorphisms. Further, we prove soundness: two pictures P and P' denote the same relation whenever there are homomorphisms from P to P' and from P' to P, which is precisely when the pictures are provably equivalent using the axioms of a upa. Since any upa has a faithful representation in a power of **Rel**, the allegory of sets [3, 5], we obtain a completeness theorem stating that two pictures are provably equivalent if and only if, for every interpretation of their basic components, they denote the same relation.

Soundness and completeness show that we can rely on pictures when deriving circuits. In the appendix, we picture an abstract specification of an adder, which cannot be implemented directly, and apply equivalences of pictures to derive a picture of a ripple adder, which is implementable using logical components. We also give a small portion of the corresponding RUBY derivation: the full derivation is explained in ten pages in [7]. By soundness, our pictorial derivation demonstrates that the two circuits compute the same relation. The pictorial derivation is much briefer than the term derivation, because each pictorial equivalence represents a long sequence of equivalences on terms. In doing the derivation, we use four additional axioms which reflect the meaning of addition. It is a feature of pictorial derivations that we can concentrate on the steps directly involving the semantics of components, as pictures abstract conveniently from repetitive applications of structural rules such as associativity.

This example demonstrates the advantages of using pictures for doing derivations. The pictures are easy to read and their structure often suggests a suitable strategy for derivations. By contrast, the corresponding RUBY terms are large and difficult to read. Scanning the pictorial derivation makes clear why the circuits are equivalent and where the real work lies in proving this: to learn this information from the RUBY derivation requires careful study. Our direct correspondence between pictures and terms allows the user to think in pictures while a machine manipulates the terms.

The notion of a upa is intermediate between an abstract allegory **A** and the regular category generated by **A**. A upa is of interest because it provides a sound interpretation of the axioms we have given, and embeds faithfully in a power of the allegory of sets. This embedding provides our completeness result. It might appear more natural to study the slightly stronger notion of unitary tabular allegory, since this corresponds to the same fragment of logic (conjunction and first order existential quantification) while coinciding with the notion of category of relations of a regular category. However, completing a pre-tabular allegory **A** to a tabular allegory involves adjoining an object for each non-maximal arrow of **A**. We view objects as types (including the natural numbers and booleans) and arrows as circuits which compute non-deterministic programs. Completion to a tabular allegory adds a type corresponding to each recursively enumerable function. These functions interest us as programs, but not as types: it is therefore appropriate to work in the pre-tabular allegory rather than in its tabular completion.

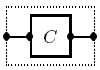
In Section 2, we introduce our calculus of pictures of circuits. In Section 3 we present the allegorical axioms as an equivalence relation on pictures. In Section 4 we give the additional axioms under which our allegory of pictures is a upa in which every type is inhabited. In Section 5 we illustrate the expressive power of our language by picturing terms expressing parallel composition, bifurcation and interchange of wires, and feedback loops. In Section 6 we define the connectivity network underlying a picture. In Section 7 we give a relational semantics for pictures and networks. In Section 8 we give an appealing interpretation to picture homomorphisms as "addition of solder". Section 9 proves the important soundness and completeness results. Section 10 indicates future areas of research. In the Appendix we present a simple example of the derivation of an implementable circuit from a high level specification.

2: Pictures

A *picture* is a graphical representation of the relationship between a given collection of basic components (*cells*) and their external pins (*connectors*) as specified by the (finite, piecewise linear) wires used to connect the various pins of the components. A picture determines a relation between its input type and its output type thus: any two points on the same wire are constrained to carry equal values, while components impose more complex constraints—for example, an `and` gate constrains its output to be the logical and of its inputs. The notions of input (left hand) and output (right hand) pins, and the consequent notion of causality, are conventional: taking a relational rather than a functional view of circuits, we consider information to flow in more than one direction in a circuit. We write U for the empty list of connectors (the unit type).

The meaning of pictures is intuitive: we omit the formal inductive definition and merely indicate the nature of the picture constructors.[1] A *picture* is either

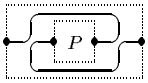
- the picture $C: A \to B$ of a cell



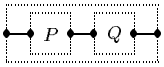
- or the picture of a wiring primitive (we often omit the bounding box)

$\mathrm{id}_A \qquad u_A \qquad \pi_{0,A,B} \quad \pi_{1,A,B}$

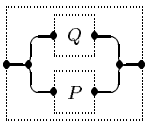
- or the *reciprocation* $P^\circ: B \to A$ of a picture $P: A \to B$ thus:



- or the *sequential composition* $P\,;Q: A \to C$ of pictures $P: A \to B$ and $Q: B \to C$ thus:



- or the *intersection* $P \cap Q: A \to B$ of pictures $P: A \to B$ and $Q: A \to B$ thus:



3: Axioms for pictures

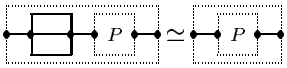
Many different pictures can express a given set of the connections between components of a circuit. In Sections 3 and 4 we define an equivalence relation $\simeq$ on pictures which is such that two pictures are equivalent if and only if they represent mutually homomorphic networks: $\simeq$ captures the transformations of pictures which do not affect connections. This leads naturally to Theorem 15, which shows that equivalent pictures denote the same relation between their input and output types.

By definition, a picture determines a term over

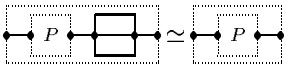
$$P ::= C \mid W \mid P^\circ \mid P\,;P \mid P \cap P,$$

where C ranges over a given set of cells and W over wiring cells. We now present our axioms for equality of pictures and further, express these axioms as equalities on terms. The equations of this section are precisely those of an allegory: thus we can define an allegory $\mathbf{P}$ in which objects are types and arrows are pictures. In Section 4 we present the additional axioms for wiring cells, which make our allegory of pictures a upa. We write $\simeq$ for the equivalence generated by the axioms of this section together with those of Section 4. The partial order $\subseteq$ on terms obtained by writing $P \subseteq Q$ for $P \cap Q \simeq P$ is a pre-congruence.

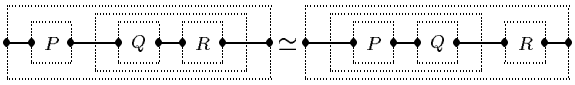
For any pictures P, Q, R of appropriate type,



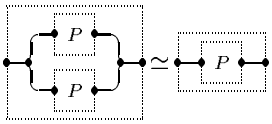
$\mathrm{id}_A\,;P \simeq P$



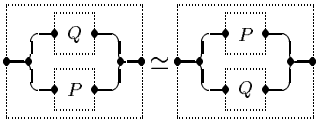
$P\,;\mathrm{id}_B \simeq P$



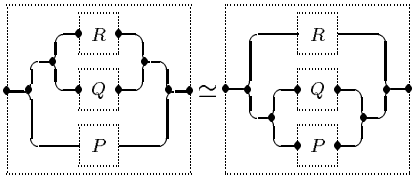
$P\,;(Q\,;R) \simeq (P\,;Q)\,;R$



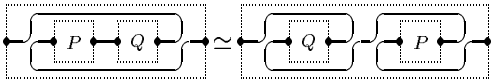
$P \cap P \simeq P$



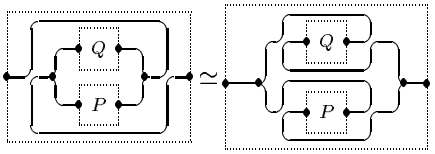
$P \cap Q \simeq Q \cap P$



$P \cap (Q \cap R) \simeq (P \cap Q) \cap R$

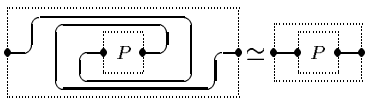


$(P\,;Q)^\circ \simeq Q^\circ\,;P^\circ$

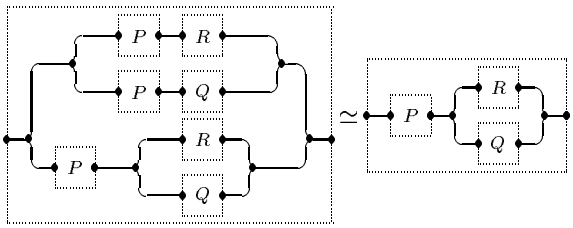


$(P \cap Q)^\circ \simeq P^\circ \cap Q^\circ$

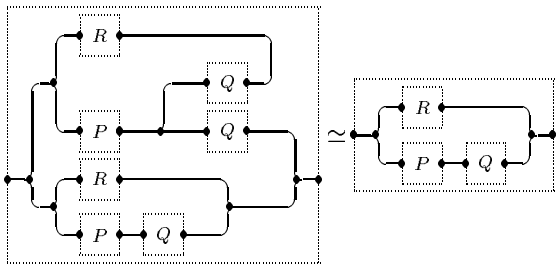
[1] Note that wires are read from bottom to top of a picture, and type information is usually omitted when we draw pictures.



$(P^\circ)^\circ \simeq P$

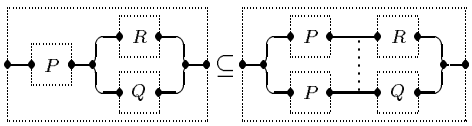


$(P\,;(Q \cap R)) \cap ((P\,;Q) \cap (P\,;R)) \simeq P\,;(Q \cap R)$

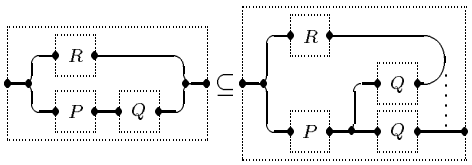


$((P\,;Q) \cap R) \cap (P \cap (R\,;Q^\circ))\,;Q) \simeq (P\,;Q) \cap R$

Remark 1 *The last two axioms above are equivalent to the inclusions* $P\,;(Q \cap R) \subseteq (P\,;Q) \cap (P\,;R)$ *and* $(P\,;Q) \cap R \subseteq (P \cap (R\,;Q^\circ))\,;Q$*, depicted thus:*



$P\,;(Q \cap R) \subseteq (P\,;Q) \cap (P\,;R)$



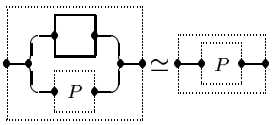
$(P\,;Q) \cap R \subseteq (P \cap (R\,;Q^\circ))\,;Q$

The first inclusion imposes monotonicity of composition in its first argument (monotonicity in the second argument being derivable); the second is the modular law. Each dotted wire above represents the additional wire whose insertion renders the two pictures equivalent (since $\cap$ *is idempotent). In Section 8 we show that, for each of these inclusions, there is a homomorphism from the picture without the dotted wire to the picture with it. Thus these pictures give a natural interpretation to two rather complex axioms.*

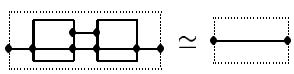
All the axioms apart from these last two and idempotence of meet can be seen as continuous deformations of pictures which preserve the connections made by wires: that is, these axioms correspond to a certain class[2] of homotopies in $\mathbb{R}^3$. Thus pictures offer a natural insight into the allegorical axioms, and make certain axioms easier to remember and apply.

4: Axioms for Rewiring Pictures

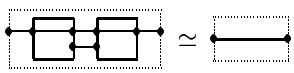
As well as basic components, we use the *wiring cells*, id_A, u_A, $\pi_{0,A,B}$, $\pi_{1,A,B}$ to effect connections among the wires of a picture. We abbreviate $u_A\,;u_B^\circ$ to $\top_{A,B}$. We require wiring cells to satisfy the following axioms:



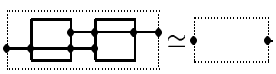
$P \cap \top_{A,B} \simeq P$



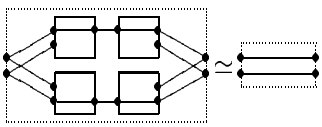
$\pi_{0,A,B}^\circ\,;\pi_{0,A,B} \simeq \mathrm{id}_A$



$\pi_{1,A,B}^\circ\,;\pi_{1,A,B} \simeq \mathrm{id}_B$



$\pi_{0,A,B}^\circ\,;\pi_{1,A,B} \simeq \top_{A,B}$



$(\pi_{0,A,B}\,;\pi_{0,A,B}^\circ) \cap (\pi_{1,A,B}\,;\pi_{1,A,B}^\circ) \simeq \mathrm{id}_{A\times B}$

We impose one other axiom, so simple we omit the corresponding picture. We require that $\mathrm{id}_U \simeq u_U$. This reflects the fact that id_U and u_U both correspond to the empty picture, with no wires or connectors. In

[2]Our homotopies permit wires to cross, and so take place in three dimensions: however, the boxes depicting cells may only be translated in the plane of the picture or rotated about an axis perpendicular to that plane. Each such homotopy of pictures corresponds to an α-equivalence of networks.

the presence of the first axiom pictured, $\mathrm{id}_U \simeq u_U$ is equivalent to the requirement that id_U be the maximal endomorphism on U.

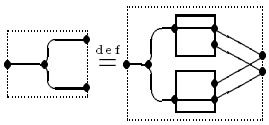
The first wiring axiom pictured states that $\top_{A,B}$ is the top element of the homposet $\mathbf{P}(A,B)$, and implies that $(u_A \,;\, u_A^\circ) \cap \mathrm{id}_A \simeq \mathrm{id}_A$, whence u_A is entire. Thus U is a *unit* [5]. The second axiom implies that $\pi^\circ_{0,A,B} \,;\, \pi_{0,A,B} \subseteq \mathrm{id}_A$, that is, $\pi_{0,A,B}$ is *simple*. The third axiom implies that $\pi_{1,A,B}$ is simple. Since $\mathrm{id}_A \subseteq \pi^\circ_{0,A,B} \,;\, \pi_{0,A,B}$, every type is inhabited and $\mathbf{P}$ is *well-supported*. The last axiom implies that $\mathrm{id}_{A\times B} \subseteq \pi_{0,A,B} \,;\, \pi^\circ_{0,A,B}$ whence $\pi_{0,A,B}$ is *entire* and thus a *map*. Similarly, $\pi_{1,A,B}$ is a map. By the last two axioms, the maps $\pi_{0,A,B}$ and $\pi_{1,A,B}$ *tabulate* $\top_{A,B}$. Since the maximum morphism in each homset of $\mathbf{P}$ is tabulated, $\mathbf{P}$ is *pretabular*.

Thus the allegory of pictures is a well-supported, unitary pretabular allegory as defined in [5]. Indeed, $\mathbf{P}$ is the free such allegory on the components C [3].

If we allow empty types, the second and third wiring axioms above become inequalities. For example, we have inequality in the second axiom only if A is inhabited and B is not, in which case $\pi^\circ_{0,A,B} ; \pi_{0,A,B}$ is empty. Empty programs (such as $\mathrm{id} \cap \mathsf{not}$ on booleans) correspond to a *short circuits* [3], and are always permitted. The assumption that types are inhabited is not fundamental (it is dropped in [3]), but reflects programming practice in RUBY, where the user is assumed to recognise and avoid short circuits.

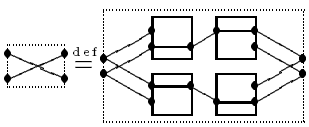
5: Derived Operations on Pictures

The calculus of pictures has many useful derived program constructors. As examples, we now define bifurcation and swapping of wires, parallel composition and feedback loops. The bifurcation operator $\mathsf{fork}_A : A \to A \times A$ that duplicates its input is defined as follows:



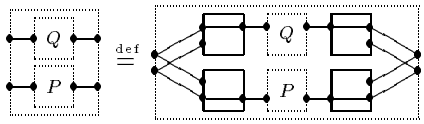
$$\mathsf{fork}_A \stackrel{\text{def}}{=} \pi^\circ_{0,A,A} \cap \pi^\circ_{1,A,A}$$

The operator $\mathsf{swap}_{A,B} : A \times B \to B \times A$ that interchanges wires is defined by:



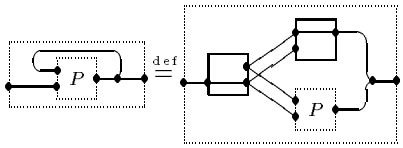
$$\mathsf{swap}_{A,B} \stackrel{\text{def}}{=} (\pi_{1,A,B} \,;\, \pi^\circ_{0,A,B}) \cap (\pi_{0,A,B} \,;\, \pi^\circ_{1,A,B})$$

The *parallel composition* (or product) $P \times Q : A \times B \to C \times D$ of pictures $P : A \to C$ and $Q : B \to D$ is defined as follows:



$$P \times Q \stackrel{\text{def}}{=} (\pi_{0,A,B} \,;\, P \,;\, \pi^\circ_{0,C,D}) \cap (\pi_{1,A,B} \,;\, Q \,;\, \pi^\circ_{1,C,D})$$

Finally, we illustrate a simple feedback loop. If $P : A \times B \to B$ then $\mathsf{feed}(P) : A \to B$ is defined by:



$$\mathsf{feed}(P) \stackrel{\text{def}}{=} \pi^\circ_{0,A,B} \,;\, (P \cap \pi_{1,A,B})$$

6: Networks

Many different pictures represent the same relationship between components and their connectors. We now abstract away from geometric layout and define the network of connections underlying a picture: we consider only the cells and external connectors of a picture and how they are connected. Such an abstraction facilitates proofs about properties common to equivalent pictures, including the program or relation they denote. We assume a collection of typed *connector names*, closed under pairing, and such that there is a countably infinite set of names of each basic type. A connector name is *basic* if it is not equal to any pair (x, y) of connector names. A name is of basic type if and only if it is a basic name. The unit type U has a unique name $*$.

Definition 2 *A* node *of type $A \to B$ is a triple (R, x, y) where $R : A \to B$ is a cell, and $x : A$ and $y : B$ are names for the connectors on the left and right faces of the cell.*

Definition 3 *A* network *of type $A \to B$ is a triple (N, l, r) where N is a finite set of nodes, and $l : A$ and $r : B$ are names. We call l and r the input and output names of the network respectively.*

Networks generalise the *net list* model of circuit connectivity, used in circuit extraction [6] and simulation [13]. A net list is a network together with geometric information about the size and position of each instance of a cell.

Remark 4 *We do not distinguish between α-equivalent networks (networks which are equal up to a type-preserving bijection on basic names).*

Definition 5 *Let (N,l,r) be a network of type $A \to B$.*

- $(N,l,r)^\circ \stackrel{\text{def}}{=} (N,r,l)$ *is a network of type* $B \to A$.
- *If (M,l',r') is a network of type $B \to C$ and ρ is the evident extension to networks of a* maximally general unifier *(MGU)*[3] *of the names r and l' then $(N,r,l)\,;(M,l',r') \stackrel{\text{def}}{=} (\rho(N \cup M), \rho l, \rho r')$ is a network of type $A \to C$.*
- *If (L,a,b) is a network of type $A \to B$ and η is an MGU of (a,b) and (l,r) then $(N,l,r) \cap (L,a,b) \stackrel{\text{def}}{=} (\eta(N \cup M), \eta l, \eta r)$ is a network of type $A \to B$.*

Each picture represents a network in an evident way:

Definition 6 *A picture P* represents *a network n if n is the image of P under the homomorphic extension to pictures of the function which maps*

- *wiring cells id_A, u_A, $\pi_{0,A,B}$ and $\pi_{1,A,B}$ to networks $(\emptyset,a,a)$, $(\emptyset,a,*)$, $(\emptyset,(a,b),a)$ and $(\emptyset,(a,b),b)$ respectively (where $a:A$ and $b:B$)*
- *and any other cell $C\colon A \to B$ to the network $(\{(C,a,b)\},a,b)$, where each basic name in (a,b) occurs only once.*

7: The Relational Semantics of Pictures and Networks

We now give a semantics to pictures and networks in terms of relations between sets. Our translation from pictures to networks respects these semantics in the sense that the same relation is denoted by a network and by any picture which represents that network.

We assume an interpretation $[\![-]\!]$ of basic types as sets, extended homomorphically to tuples of types (with $[\![U]\!] = \{*\}$), and assume for each cell $C\colon A \to B$ a binary relation $[\![C]\!] \subseteq [\![A]\!] \times [\![B]\!]$. The wiring cells have their standard set-theoretic interpretation: for example, $[\![\mathrm{id}_A]\!] = \{(a,a) \mid a \in [\![A]\!]\}$. We extend $[\![-]\!]$ homomorphically from pictures of cells to pictures, via the usual set-theoretic definitions of reciprocation, composition, and intersection on relations. Given a set V of basic names, a valuation θ of V assigns an element of $[\![A]\!]$ to each $x \in V$ of type A. We extend θ homomorphically to a valuation of names with base names in V.

Definition 7 *Let (N,l,r) be a network of type $A \to B$ with basic connector names V. Then (N,l,r) denotes the relation $\ulcorner(N,l,r)\urcorner$ given by:*

$$\{(a,b) \in [\![A]\!] \times [\![B]\!] \mid \text{there is a valuation } \theta \text{ of } V \text{ such that } \theta l = a,\ \theta r = b \text{ and if } (C,x,y) \in N \text{ then } (\theta x, \theta y) \in [\![C]\!]\}.$$

Lemma 8 *For any networks $n = (N,l,r)$ and $m = (M,l',r')$, of appropriate type, $\ulcorner n^\circ \urcorner = \ulcorner n \urcorner^\circ$, $\ulcorner n\,;m \urcorner = \ulcorner n \urcorner ; \ulcorner m \urcorner$ and $\ulcorner n \cap m \urcorner = \ulcorner n \urcorner \cap \ulcorner m \urcorner$.*

Proposition 9 *If a picture P represents the network n then $[\![P]\!] = \ulcorner n \urcorner$.*

8: Homomorphisms on pictures

Definition 10 *Let $n = (N,a,b)$ and $m = (M,c,d)$ be networks of the same type. A* homomorphism *$f\colon n \to m$ is a type-preserving function f mapping basic connector names in n to basic connector names in m, extended homomorphically to compound connector names, such that*

- *$f(a) = c$ and $f(b) = d$,*
- *if $(C,l,r) \in N$ then $(C, f(l), f(r)) \in M$.*

A homomorphism between networks corresponds to an inclusion of the relations they denote, thus:

Lemma 11 *If there is a homomorphism of networks $f\colon n \to m$ then $\ulcorner m \urcorner \subseteq \ulcorner n \urcorner$.*

Definition 12 *Let $P : A \to B$ and $Q : A \to B$ be pictures. Let I_P and O_P be respectively the lists of input and output connectors of P, in ascending order, and similarly I_Q and O_Q. A* homomorphism *from P to Q is a type-preserving function h from connectors of P to connectors of Q such that*

- *$h(I_P) = I_Q$, $h(O_P) = O_Q$ and*
- *for each cell C pictured in P with inputs I_C and outputs O_C, there is a cell C pictured in Q with inputs $h(I_C)$ and outputs $h(O_C)$.*

We write $P \rightleftharpoons Q$ if there are homomorphisms $h\colon P \to Q$ and $k\colon Q \to P$.

Proposition 13 *The category of pictures and picture homomorphisms is equivalent to the category of networks and network homomorphisms.*

A picture homomorphism $h\colon P \to Q$ has a very natural interpretation. Every cell of P maps to the same cell in Q: further, h may connect wires which are distinct in P. Thus Q can be obtained from P by "adding solder", that is, by connecting wires which are not connected in P and by wiring new components into P. This process is illustrated by the inequational forms of the monotonicity and modular laws given in Remark 1. The right hand picture in each case has a

[3] that is, ρ is an idempotent substitution on basic names such that $\rho(r) = \rho(l')$ and any other substitution ρ' such that $\rho'(r) = \rho'(l')$ is equal to $\sigma \circ \rho$ for some substitution σ. The typing restriction for composition ensures that such an MGU exists.

dotted wire. There is a homomorphism from the picture without the wire to the picture with the wire, which at the network level identifies the names of the two wires linked by the dotted wire. In each case the image of the homomorphism is provably equivalent to the left hand picture, by idempotence of $\cap$.

Each cell in a picture imposes a constraint on the values of its input and output wires, while connecting two wires forces them to carry the same value. Thus if $h: P \to Q$ then Q imposes more constraints on the values carried by its wires than P: this is reflected by the following corollary of Lemma 11:

Corollary 14 *If $h: P \to Q$ is a picture homomorphism then $[\![Q]\!] \subseteq [\![P]\!]$. If $P \rightleftharpoons Q$ then $[\![P]\!] = [\![Q]\!]$.*

9: Soundness and completeness

We now present our main result, that two pictures can be proved equal using the axioms for a upa if and only if each is homomorphic to the other. By Corollary 14, they are equivalent only if they denote the same relation. In fact, by the proof of Theorem 15, two pictures are provably equivalent if and only if they denote the same relation under any interpretation of their basic components. These justify our use of the upa axioms in deriving programs like the ripple adder from the specification in Figure 1 (see Appendix).

Theorem 15 *Let P and Q be pictures. $P \simeq Q$ if and only if $P \rightleftharpoons Q$.*

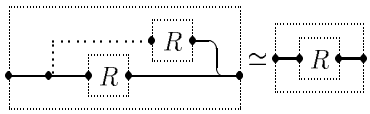
Proof: For the forward implication, as both $\simeq$ and $\rightleftharpoons$ are congruences, it suffices to prove the result when $P \simeq Q$ is an axiom instance. For each of the axioms of Sections 3 and 4 except monotonicity of composition and the modular law, the pictures involved denote α-equivalent networks, and so are isomorphic. In the remaining two cases, homomorphisms are readily found between the networks representing the pictures concerned, as indicated in Remark 1.

Conversely, by Corollary 14, if $P \rightleftharpoons Q$ then $[\![P]\!] = [\![Q]\!]$, whatever the interpretation of the basic cells of P. A upa has a faithful representation [5] in a power of **Rel**. Hence if $[\![P]\!] = [\![Q]\!]$ for any interpretation of the basic components of P, then [3] $P \simeq Q$. □

Example 1 *Let $R: A \to B$ and consider the equation $R\,;\mathsf{rng}(R) \simeq R$, where $\mathsf{rng}(R) \stackrel{\text{def}}{=} id_B \cap (\top_{B,A}\,;R)$ is the identity on the range of R. The equation can be proved using the allegorical axioms:*

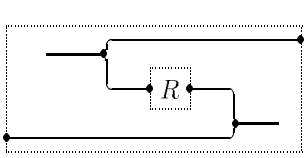
$$\begin{aligned} R\,;(id_B \cap (\top_{B,A}\,;R)) &\subseteq (R\,;id_B) \cap (R\,;\top_{B,A}\,;R) \\ &\subseteq R\,;id_B \subseteq R \qquad \textit{and} \\ R\,;(id_B \cap (\top_{B,A}\,;R)) &\supseteq R\,;(id_B \cap (R^\circ\,;R)) \\ &\supseteq R \cap (R\,;id_B) \supseteq R \end{aligned}$$

Alternatively, the equivalence follows by Theorem 15, observing that the terms represent the nets $(\{(R,x,y),(R,z,y)\},x,y)$ and $(\{(R,a,b)\},a,b)$, and these networks are mutually homomorphic via $\{x \mapsto a, y \mapsto b, z \mapsto a\}$ and $\{a \mapsto x, b \mapsto y\}$. Mapping both x and z to a corresponds to adding a (dotted) wire to make the pictures of the two terms equivalent:



This wire witnesses the fact that there is a homomorphism from the left hand picture to the right hand picture (add the dotted wire and the identify the two copies of R). There is an evident inclusion of the right hand picture in the left.

Example 2 *Pictures illuminate the isomorphism between the category of small upas and the category of small discrete cartesian bicategories (dcbs). To prove these two categories isomorphic we define the structure of one in terms of the structure of the other, and conversely, and prove these definitions mutually inverse. As an example, if we translate R° into a dcb and then back to a upa, we obtain the term $\pi^\circ_{0,B,A}\,;(id_B \times (\mathsf{fork}_A\,;(R \times id_A)))\,;\alpha_{B,B,A}\,;(\mathsf{fork}^\circ_B \times id_A)\,;\pi_{1,B,A}$, with picture:*



The complex picture makes the same connections as the simpler picture given in Section 2, and so their networks are mutually homomorphic. By Proposition 13 the pictures themselves are mutually homomorphic, and by Theorem 15, the pictures are equivalent. Furthermore, by Corollary 14 the pictures denote the same set theoretic relation. Notice how much easier it is to construct the picture of R° than to construct the corresponding term.

10: Future Work

Our results have several natural extensions. In order to design grid-like circuits where cells and pictures have connectors on all four sides we might consider a suitable notion of double allegory. This relates to work by Molitor [11] in which circuits can be composed either at east-west or at north-south interfaces. We are also considering how to model RUBY's treatment of clocked circuits, and how to augment our axioms with new equations, as is done informally in the appendix.

Appendix: Using Pictures to Guide Derivations

We derive a ripple adder from a high level specification. We assume a type $\mathbb{B}$ of bits (*false* and *true*) and $\mathbb{N}$ of natural numbers, together with primitives $\beta\colon \mathbb{B} \to \mathbb{N}$, $*2\colon \mathbb{N} \to \mathbb{N}$, and $+\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, with interpretations

$$\begin{aligned} [\![\beta]\!] &= \{(\mathit{false}, 0), (\mathit{true}, 1)\} \\ [\![*2]\!] &= \{(x, 2x) \mid x \in \mathbb{N}\} \\ [\![+]\!] &= \{((x, y), x + y) \mid x, y \in \mathbb{N}\end{aligned}$$

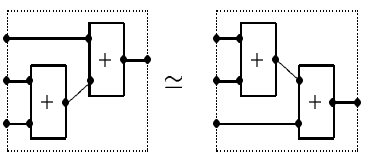
A program $\mathit{eval}_3\colon (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \to \mathbb{N}$ that converts a 3-bit binary number to a natural number in the range 0–7 is defined by (writing $*2^2$ for $*2\,;*2$):

$$\mathit{eval}_3 \stackrel{\text{def}}{=} (((\beta \times \beta) \times \beta)\,;((*2^2 \times *2) \times \mathrm{id}_{\mathbb{N}})\,;(+ \times \mathrm{id}_{\mathbb{N}})\,;+.$$

We define eval_n analogously for any positive integer n. A circuit add (pictured in Figure 1) which on input of a 3-bit binary number and a bit, adds them and outputs the result as a 4-bit binary number, is specified by the term $(\mathit{eval}_3 \times \beta)\,;+\,;\,\mathit{eval}_4{}^\circ$.

The term add expresses that a bit can be added to a binary number by converting the binary number and bit to natural numbers, adding them, and converting the result back to binary. We cannot implement add directly in hardware as it is defined using arithmetic primitives rather than logical primitives, and contains non-deterministic components such as $+^\circ$.

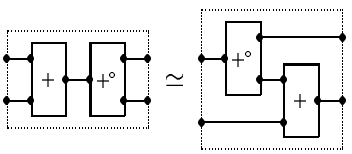
The first step in deriving an implementable circuit for add uses the associativity of $+$. Writing α for the natural iso $\alpha_{A,B,C}\colon (A \times B) \times C \to A \times (B \times C)$, definable using projections, we apply the additional axiom:



$$(+ \times \mathrm{id})\,;+ \;\simeq\; \alpha\,;(\mathrm{id} \times +)\,;+$$

to transform the upwards staircase of $+$s in $(\mathit{eval}_3 \times \beta)$ into a downwards staircase, as in Figure 2. A corresponding lengthy calculation shows that (omitting type subscripts) add $\simeq \alpha\,;\alpha\,;(\beta \times (\beta \times (\beta \times \beta)))\,;(*2^2 \times (*2^1 \times (*2^0 \times \mathit{id})))\,;(\mathrm{id} \times ((\mathrm{id} \times +)\,;+))\,;+\,;\,\mathit{eval}_4{}^\circ$.

We now use the following property of $+$

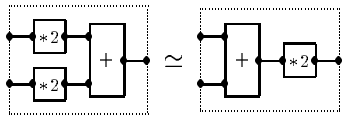


$$+\,;+^\circ \simeq (\mathrm{id} \times +^\circ)\,;\alpha^\circ\,;(+ \times \mathrm{id})$$

to slide the $+$ and $+^\circ$ staircases past one another to form a single $+\,;+^\circ$ staircase, as in Figure 3. The corresponding calculation (omitting subscripts) is

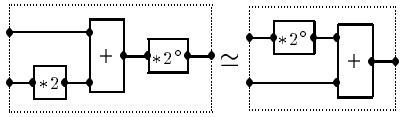
$$\begin{aligned} & (\mathrm{id} \times ((\mathrm{id} \times +)\,;+))\,;+\,;+^\circ; \\ & \qquad (+^\circ \times ((+^\circ\,;(+^\circ \times \mathrm{id})) \times \mathrm{id}) \\ \simeq\ & (\mathrm{id} \times ((\mathrm{id} \times +)\,;+\,;+^\circ))\,;\alpha^\circ; \\ & \qquad ((+\,;+^\circ\,;(+^\circ \times \mathrm{id})) \times \mathrm{id}) \\ \simeq\ & (\mathrm{id} \times ((\mathrm{id} \times (+\,;+^\circ))\,;\alpha^\circ\,;(+ \times \mathrm{id})))\,;\alpha^\circ; \\ & \qquad (((\mathrm{id} \times +^\circ)\,;\alpha^\circ\,;((+\,;+^\circ) \times \mathrm{id})) \times \mathrm{id}) \\ \simeq\ & (\mathrm{id} \times ((\mathrm{id} \times (+\,;+^\circ))\,;\alpha^\circ\,;((+\,;+^\circ) \times \mathrm{id})))\,;\alpha^\circ; \\ & \qquad ((\alpha^\circ\,;((+\,;+^\circ) \times \mathrm{id})) \times \mathrm{id}) \end{aligned}$$

We next combine distributivity of $*2$ over $+$:



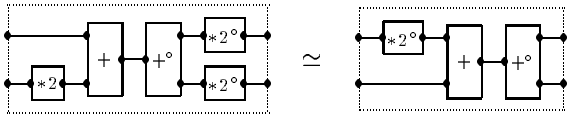
$$(*2 \times *2)\,;+ \;\simeq\; +\,;*2$$

with the equation relating two terms describing a $+$ cell in which the second input is constrained to be even:



$$(*2 \times \mathrm{id})\,;+\,;*2^\circ \simeq (\mathrm{id} \times *2^\circ)\,;+$$

to obtain the derived equation:



$$(*2 \times \mathrm{id})\,;+\,;+^\circ\,;(*2^\circ \times *2^\circ) \simeq (\mathrm{id} \times *2^\circ)\,;+\,;+^\circ$$

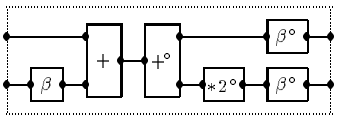
As a RUBY theorem this is called Horner's rule, being a natural generalisation of Horner's rule for evaluating polynomials efficiently. The resulting picture is given in Figure 4: the corresponding calculation is:

$$\begin{aligned} & (*2^2 \times (*2^1 \times (*2^0 \times \mathrm{id})))\,;(\mathrm{id} \times ((\mathrm{id} \times (+\,;+^\circ)); \\ & \qquad \alpha^\circ\,;((+\,;+^\circ) \times \mathrm{id})))\,;\alpha^\circ\,;((\alpha^\circ\,;((+\,;+^\circ) \times \mathrm{id})) \\ & \qquad \times \mathrm{id})\,;(((*2^{3^\circ} \times *2^{2^\circ}) \times *2^{1^\circ}) \times *2^{0^\circ}) \\ \simeq\ & (\mathrm{id} \times ((\mathrm{id} \times (+\,;+^\circ))\,;\alpha^\circ\,;(((*2 \times \mathrm{id})\,;+\,;+^\circ; \\ & \qquad (\mathrm{id} \times *2^\circ)) \times \mathrm{id})))\,;\alpha^\circ\,;((\alpha^\circ\,;((((*2\,;*2) \times \mathrm{id})\,;+; \\ & \qquad +^\circ\,;((*2^\circ\,;*2^\circ\,;*2^\circ) \times (*2^\circ\,;*2^\circ))) \times \mathrm{id})) \times \mathrm{id}) \\ \simeq\ & (\mathrm{id} \times ((\mathrm{id} \times (+\,;+^\circ))\,;\alpha^\circ\,;(((*2 \times \mathrm{id})\,;+\,;+^\circ; \\ & \qquad (*2^\circ \times *2^\circ)) \times \mathrm{id})))\,;\alpha^\circ\,;((\alpha^\circ\,;(((*2 \times \mathrm{id}); \\ & \qquad +\,;+^\circ\,;((*2^\circ\,;*2^\circ) \times *2^\circ)) \times \mathrm{id})) \times \mathrm{id}) \\ \simeq\ & (\mathrm{id} \times ((\mathrm{id} \times (+\,;+^\circ\,;(*2^\circ \times \mathrm{id})))\,;\alpha^\circ; \\ & \qquad ((+\,;+^\circ\,;(*2^\circ \times \mathrm{id})) \times \mathrm{id})))\,;\alpha^\circ; \\ & \qquad ((\alpha^\circ\,;((+\,;+^\circ\,;(*2^\circ \times \mathrm{id})) \times \mathrm{id})) \times \mathrm{id}) \end{aligned}$$

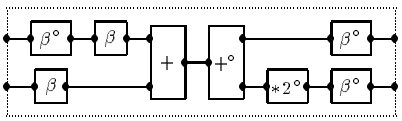
We have now shown that add $\simeq \alpha\,;\alpha\,;(\beta \times (\beta \times (\beta \times \beta)))\,;(\mathrm{id} \times ((\mathrm{id} \times (+\,;+^\circ\,;(*2^\circ \times \mathrm{id})))\,;\alpha^\circ\,;((+\,;+^\circ\,;(*2^\circ \times \mathrm{id})) \times \mathrm{id})))\,;\alpha^\circ\,;((\alpha^\circ\,;((+\,;+^\circ\,;(*2^\circ \times \mathrm{id})) \times \mathrm{id})) \times \mathrm{id})\,;(((\beta^\circ \times \beta^\circ) \times \beta^\circ) \times \beta^\circ)$.

Next we address the type conversions β. The following equation expresses that if the lower input and the outputs of $+\,;+^\circ\,;(*2^\circ \times \mathrm{id})$ are constrained to be

in $\{0,1\}$, then so is the upper input:

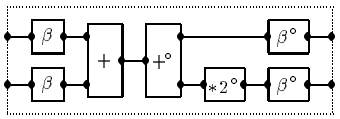


$$(\beta \times \mathrm{id})\,;+\,;+^\circ\,;(*2^\circ\,;\beta^\circ)\times\beta^\circ$$

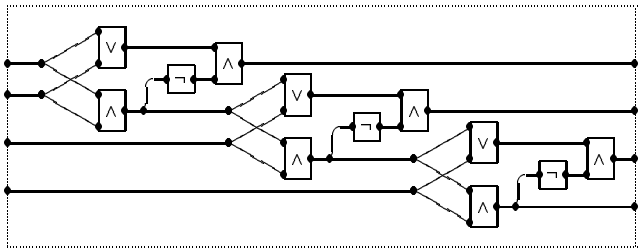


$$\beta\times(\beta^\circ\,;\beta)\,;+\,;+^\circ\,;(*2^\circ\,;\beta^\circ)\times\beta^\circ$$

Applying this result twice to the adder yields Figure 5. Finally, we observe that the component



takes two logical values as input, and outputs their logical carry and sum. This is the semantics of a *half adder* ha, which can be implemented using standard logical components: $\mathsf{ha} \stackrel{\text{def}}{=} \langle \mathsf{and}, \mathsf{or}\rangle\,;(\mathsf{fork}\times\mathrm{id})\,;\alpha\,;(\mathrm{id}\times((\mathsf{not}\times\mathrm{id})\,;\mathsf{and}))$. In conclusion, we have derived the standard ripple-adder implementation for add, given by the term $\mathsf{add} = \alpha\,;\alpha\,;(\mathrm{id}\times((\mathrm{id}\times\mathsf{ha})\,;\alpha^\circ\,;(\mathsf{ha}\times\mathrm{id})))\,;\alpha\,;((\alpha\,;(\mathsf{ha}\times\mathrm{id}))\times id)$:



Acknowledgements

The ideas in this paper benefited greatly from discussions with Peter Freyd, Alan Jeffrey, Edmund Robinson and Paul Taylor. The pictures could not have been drawn without Kris Rose's help using *xypic*.

References

[1] Roland Backhouse and Ed Voermans and Jaap van der Woude, *A Relational Theory of Datatypes*, Proc. STOP Summer School on Constructive Algorithmics,, Ameland, The Netherlands, 1992.

[2] Richard Bird and Oege de Moor, *From Dynamic Programming to Greedy Algorithms*, Proc. STOP Summer School on Constructive Algorithmics, Ameland, The Netherlands, 1992.

[3] Carolyn Brown and Alan Jeffrey, *Allegories of Circuits*, to appear in Proc. Symposium on Logical Foundations of Computer Science, St Petersburg, 1994.

[4] Aurelio Carboni and Bob Walters, *Cartesian Bicategories I*, Journal of Pure and Applied Algebra 49 (1987) 11-32.

[5] Peter Freyd and Andre Scedrov, *Categories, Allegories*, North-Holland, 1990.

[6] Randall L. Geiger and Phillip E. Allen and Noel R. Strader, *VLSI Design Techniques for Analog and Digital Circuits*, McGraw Hill, 1990.

[7] Graham Hutton, *A Relational Derivation of a Functional Program*, lecture notes of the STOP Summer School on Constructive Algorithmics, Ameland, The Netherlands, 1992.

[8] Graham Hutton, *Between Functions and Relations in Calculating Programs*, PhD thesis, Univ. of Glasgow, Oct. 1992, available as Research Report FP-93-5.

[9] Graham Hutton, Erik Meijer and Ed Voermans, *A tool for relational programmers*, distributed on the MOP (mathematics of programming) mailing list, January 1994. Copies available by e-mail from graham@cs.chalmers.se.

[10] Geraint Jones and Mary Sheeran, *Designing arithmetic circuits by refinement in Ruby*, in Proc. Second International Conference on Mathematics of Program Construction, 1992, to appear in LNCS, Springer-Verlag.

[11] Paul Molitor, *Free net algebras in VLSI-theory*, Fundamenta Informaticae IX (1988) 117-142.

[12] Oege de Moor, *Categories, Relations and Dynamic Programming*, D.Phil thesis, Oxford University, 1992, available as Research Report PRG-98.

[13] Steven M. Rubin, *Computer Aids for VLSI Design*, Addison Wesley, 1987.

[14] Mary Sheeran, *Describing and reasoning about circuits using relations*, in Tucker et al., editors, Proc. Workshop in Theoretical Aspects of VLSI, Leeds, 1986.

[15] Mary Sheeran, *Retiming and slowdown in* Ruby, in Milne, editor, The Fusion of Hardware Design and Verification, North-Holland, 1988.

[16] Mary Sheeran, *Describing butterfly networks in* Ruby, in *Proc. Glasgow Workshop on Functional Programming*, 1989, S-V, Workshops in Computing.

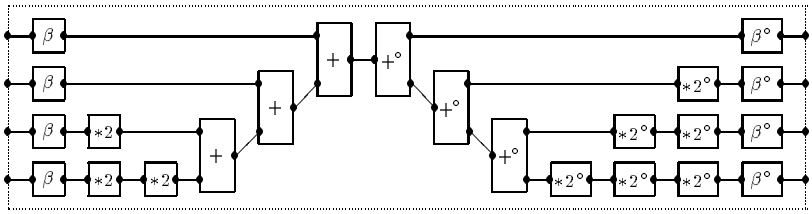


Figure 1: The picture of the specification add

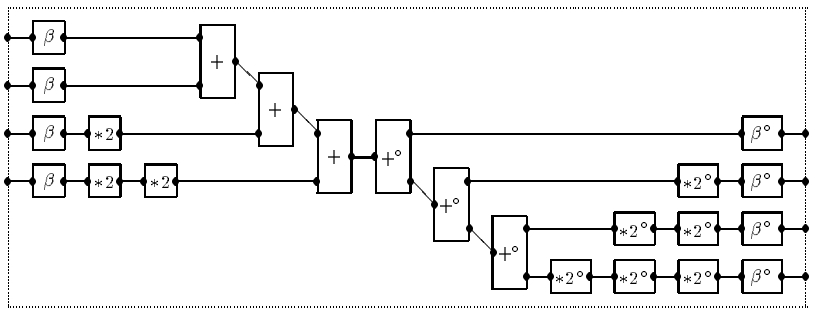


Figure 2: Circuit after applying Associativity of +

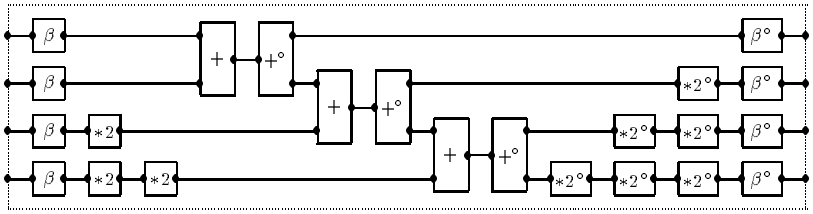


Figure 3: Circuit after conversion to a single staircase

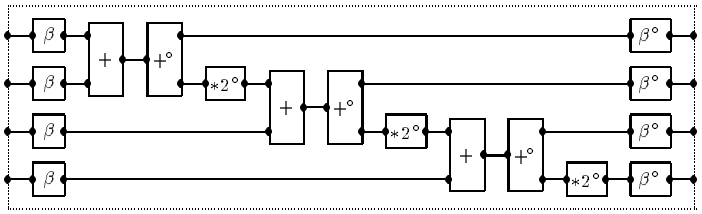


Figure 4: Circuit after applying Horner's rule

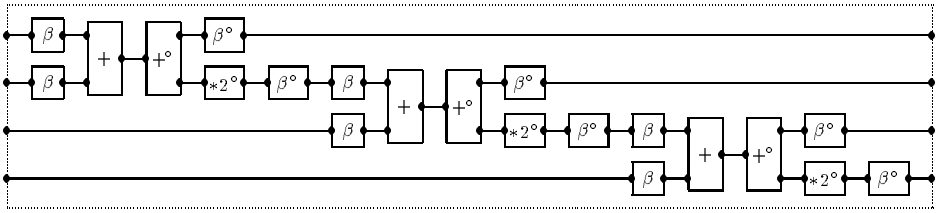


Figure 5: Circuit after applying Distributivities